# ANÁLISIS DEL RETO GRUPO 7

*Karin Guerrero, 202317306, k.guerreroh*

*Daniel Niño, 202325515, df.nino2*

## Requerimiento 1

```
def req_1(data_structs, N, country, level):
    filter_data = lt.newList('ARRAY_LIST') # O(1)
    map_by_country = data_structs['jobs_country'] # O(1)
    country_jobs = (mp.get(map_by_country, country))['value'] # O(1)

    amount_jobs_country = lt.size(country_jobs)

    for job in lt.iterator(country_jobs): # O (h) siendo h la cantidad de ofertas que haya en la ciudad
        if job['experience_level'] == level: # O(1)
            lt.addLast(filter_data, job)

    merg.sort(filter_data, sort_job_by_date) # O(klog(k)) siendo k la cantidad de ofertas que cumplan con el nivel de experiencia y el país

    amount_jobs_experience = lt.size(filter_data) # O(1)


    if N > lt.size(filter_data): # O(1)
        N = lt.size(filter_data)

    answer = lt.subList(filter_data, 1, N) # O(N) SIendo n la cantidad de datos que nos piden

    return answer, amount_jobs_country, amount_jobs_experience
```

## Descripción

El código filtra y clasifica las ofertas de trabajo. Al recibir una estructura de datos, un país específico y un nivel de experiencia, la función extrae los trabajos correspondientes al país dado y los clasifica según el nivel de experiencia proporcionado. Luego de filtrar y ordenar estos trabajos por fecha, la función devuelve los primeros “N” trabajos, ajustando “N” si es necesario para no exceder el número total de trabajos disponibles. La función también proporciona información sobre la cantidad total de trabajos en el país y la cantidad de trabajos que coinciden con el nivel de experiencia especificado.

| | |
|---|---|
| **Entrada** | Estructuras de datos, cantidad de datos, código del país y nivel de Experticia. |
| **Salidas** | Sublista de total de datos, cantidad de trabajos por país y cantidad de trabajos por nivel de experticia. |
| **Implementado (Sí/No)** | Si se implementó y Karin Guerrero. |

## Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

| **Pasos** | **Complejidad** |
|---|---|
| Paso 1 | O(1) |

| | |
|---|---|
| Paso 2 | O(1) |
| Paso 3 | O(1) |
| Paso 4 | O(N) |
| *Paso 5* | *O(1)* |
| *Paso 6* | *O(Nlog(N))* |
| *Paso 7* | *O(1)* |
| *Paso 8* | *O(1)* |
| *Paso 9* | *O(N)* |
| ***TOTAL*** | *O(N)* |

## Pruebas Realizadas

Descripción de las pruebas de tiempos de ejecución y memoria utilizada. Incluir descripción del procedimiento, las condiciones, las herramientas y recursos utilizados (librerías, computadores donde se ejecutan las pruebas, entre otros).

| | |
|---|---|
| Procesadores | Intel Core i5-1135G7 |
| Memoria RAM | 8 GB |
| Sistema Operativo | Windows 11 |

| **Entrada** | **Tiempo (s)** |
|---|---|
| small | 0.04 |
| 5 pct | 0.26 |
| 10 pct | 0.35 |
| 20 pct | 0.47 |
| 30 pct | 0.87 |
| 50 pct | 1.35 |
| 80 pct | 1.75 |
| large | 1.79 |

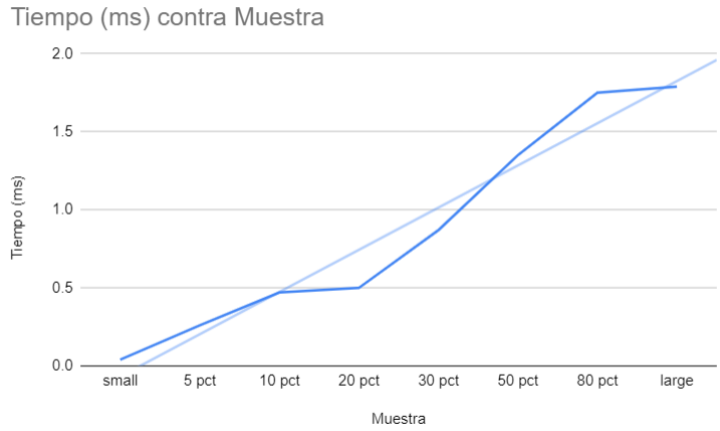
### Tablas de datos

Las tablas con la recopilación de datos de las pruebas.

| **Muestra** | **Salida** | **Tiempo (ms)** |
|---|---|---|
| small | Dato1 | 0.04 |
| 5 pct | Dato2 | 0.26 |
| 10 pct | Dato3 | 0.35 |
| 20 pct | Dato4 | 0.47 |
| 30 pct | Dato5 | 0.87 |
| 50 pct | Dato6 | 1.35 |
| 80 pct | Dato7 | 1.75 |
| large | Dato8 | 1.79 |

## Graficas

Las gráficas con la representación de las pruebas realizadas.




## Análisis

La gráfica muestra que hay una tendencia de aumento en la medida que el tamaño aumenta. Sin embargo, se observa una mayor variabilidad entre el 30 pct y el 50 pct. Además se puede observar que el tamaño más grande es ligeramente superior al valor 80 pct que seria “large”, esta diferencia es mínima, sugiriendo una posible estabilización en la medida para tamaños muy grandes.

# Requerimiento 3

```python
def req_3(control, company_name, initial_date, final_date):
    """
    Función que soluciona el requerimiento 3
    """
    # TODO: Realizar el requerimiento 3

    mapa_ofertas_empresa = control['jobs_empresa']
    valores = mp.valueSet(control['jobs_empresa'])

    lista_ofertas_empresa = me.getValue(mp.get(mapa_ofertas_empresa, company_name))


    junior = 0
    mid = 0
    senior = 0
    print(type(mapa_ofertas_empresa))

    for job in lt.iterator(valores):
        if job["elements"][0]['company_name'] == company_name and str(job["elements"][0]['published_at']) >= initial_date and str(job["elements"][0]['published_at']) <= final_date:
            lt.addLast(lista_ofertas_empresa, job)
            if job["elements"][0]['experience_level'] == 'junior':
                junior += 1
            elif job["elements"][0]['experience_level'] == 'mid':
                mid += 1
            else:
                senior += 1

    return mapa_ofertas_empresa, junior, mid, senior
```

## Descripción

Este código primero obtiene la información de un mapa guardado en una variable “mapa_ofertas_empresa” del mapa control[jobs_empresa], que contiene información sobre ofertas de trabajo por empresa. Luego, itera sobre estos valores, filtra los trabajos por empresa, fecha de publicación y nivel de experiencia y de acuerdo con esto cuenta el número de ofertas de trabajo por nivel de experiencia (junior, mid, senior). Finalmente retorna mapa_ofertas_empresa y el conteo de ofertas por nivel de experiencia.

| | |
|---|---|
| **Entrada** | Estructuras de datos, nombre de la compañía, fecha inicial y fecha final. |
| **Salidas** | Mapa con las ofertas de empresa y el conteo de experticia |
| **Implementado (Sí/No)** | Si se implementó y Karin Guerrero |

## Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

| **Pasos** | **Complejidad** |
|---|---|
| Paso 1 | O(1) |
| Paso 2 | O(1) |
| Paso 3 | O(1) |
| Paso 4 | O(1) |
| Paso 5 | O(N) |
| Paso 6 | O(1) |
| Paso 7 | O(1) |
| ***TOTAL*** | ***O(N)*** |

## Pruebas Realizadas

Descripción de las pruebas de tiempos de ejecución y memoria utilizada. Incluir descripción del procedimiento, las condiciones, las herramientas y recursos utilizados (librerías, computadores donde se ejecutan las pruebas, entre otros).

| | |
|---|---|
| Procesadores | Intel Core i5-1135G7 |
| Memoria RAM | 8 GB |
| Sistema Operativo | Windows 11 |

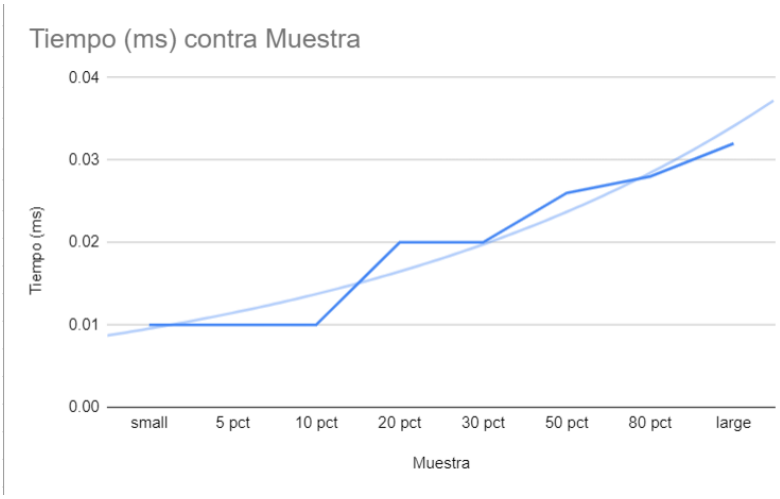
| **Entrada** | **Tiempo (s)** |
|---|---|
| small | 0.01 |
| 5 pct | 0.01 |
| 10 pct | 0.01 |
| 20 pct | 0.02 |
| 30 pct | 0.02 |
| 50 pct | 0.026 |
| 80 pct | 0.028 |
| large | 0.032 |

## Tablas de datos

Las tablas con la recopilación de datos de las pruebas.

| Muestra | Salida | Tiempo (ms) |
|---|---|---|
| small | Dato1 | 0.01 |
| 5 pct | Dato2 | 0.01 |
| 10 pct | Dato3 | 0.01 |
| 20 pct | Dato4 | 0.02 |
| 30 pct | Dato5 | 0.02 |
| 50 pct | Dato6 | 0.026 |
| 80 pct | Dato7 | 0.028 |
| large | Dato8 | 0.032 |

## Graficas




## Análisis

En esta grafica se observa una consistencia en los tiempos de ejecución para las categorías "small", "5 pct" y "10 pct", todos registrando 0.01 segundos. A medida que avanzamos hacia porcentajes mayores, los tiempos de ejecución muestran un aumento gradual, desde 0.02 segundos para "20 pct" y "30 pct", hasta 0.032 segundos para "large". Sin embargo, entre el "30 pct" y el "50 pct", hay una variación menor en el tiempo de ejecución. Además, se observa otro incremento en el tiempo de ejecución al pasar del "50 pct" al "80 pct", mostrando nuevamente un aumento gradual.

**Requerimiento 4: Daniel Felipe Niño**

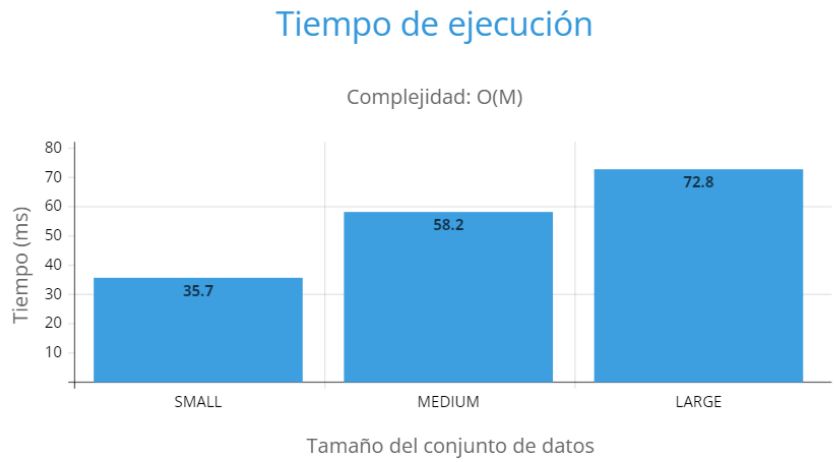
Complejidad en (O):

- Obtención del mapa de ofertas por país: Cuando se trata de simplemente acceder a una estructura de datos, tiene una complejidad de tiempo O(1).
- Obtener datos de los valores del mapa y de las ofertas en un país indicado: Al acceder a los valores de un mapa, la complejidad es O(n), donde n es el número de ofertas en el mapa. Acceder a una oferta específica en la lista *ofertas_pais* también tiene una complejidad O(n).
- Crear mapas para almacenar empresas y ofertas por ciudad: Esta operación implica crear nuevas estructuras de datos y en este caso tiene una complejidad de tiempo O(1).
- Recorrer las ofertas y conteo de empresas y ciudades indicadas: en el bucle principal de la función y su complejidad depende del número de ofertas. En el peor caso, la complejidad es O(n), donde n es el número total de ofertas.

**Tabla comparativa del experimento:**

| Tamaño del conjunto de datos | Tiempo de ejecución (ms) | Complejidad |
|---|---|---|
| Small | 35.7 | O(n) |
| Medium | 59.2 | O(n) |
| Large | 72.8 | O(n) |

**Grafica:**




**Conclusiones:**

- Como se esperaba el tiempo de ejecución aumenta juntamente con el tamaño de datos, la función tiene complejidad lineal.
- El algoritmo es eficiente en términos de ejecución.

# Requerimiento 6

```python
def req_6(control, numero_ciudad, level, year):
    """
    # TODO: Realizar el requerimiento 6
    valores = mp.valueSet(control['jobs_ciudad'])
    llaves = mp.keySet(control["jobs_ciudad"])
    mapa = mp.newMap(numelements=10,
           prime=20,
           maptype='PROBING',
           loadfactor = 1,
           cmpfunction=None)
    for job in lt.iterator(valores):
        for llave in lt.iterator(llaves):
            if job["elements"][0]["experience_level"] == level and (job["elements"][0]["published_at"])[0:4] == year:
                mp.put(mapa, llave, job)

    mapa_retorno = mp.newMap(numelements=10,
           prime=20,
           maptype='PROBING',
           loadfactor = 1,
           cmpfunction=None)
    valores2 = mp.valueSet(mapa)
    llaves2 = mp.keySet(mapa)
    n_empresas = mp.size(mapa)
    n_ciudades = lt.size(llaves)
    i = 0
    while i < int(numero_ciudad):
        inicial_size = 0
        job3 = ""
        llave3= ""
        for job in lt.iterator(valores2):
            for llave in lt.iterator(llaves2):
                size = len(job["elements"])
                if size > inicial_size:
                    inicial_size = size
                    job3 = job["elements"]
                    llave3 = llave


                pos2 = lt.isPresent(llaves2, llave3)
                lt.deleteElement(llaves2, pos2)
                mp.put(mapa_retorno, llave3, job3)

                i += 1
    return mapa_retorno, n_ciudades, n_empresas
```

## Descripción

Este código itera sobre los valores y las llaves del mapa “control['jobs_ciudad']”. Luego filtra los trabajos por nivel de experiencia y año de publicación. Crea un nuevo mapa (“mapa_retorno”) que contiene los trabajos filtrados con esto, luego calcula el número de estos y selecciona el número especificado de ciudades con la mayor cantidad de trabajos y los agrega para luego retornar el mapa de trabajos por ciudad, el número de ciudades y el número de empresas.

| | |
|---|---|
| **Entrada** | Estructuras de datos, numero de ciudad, nivel de experticia y año. |
| **Salidas** | Mapa de retorno, número de ciudades y número de empresas |
| **Implementado (Sí/No)** | Si se implementó y Karin Guerrero |

## Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

| **Pasos** | **Complejidad** |
|---|---|
| Paso 1 | O(1) |
| Paso 2 | O(1) |
| Paso 3 | O(1) |
| Paso 4 | O(n^2) |
| Paso 5 | O(1) |
| Paso 6 | O(1) |
| Paso 7 | O(m*n^2) |
| ***TOTAL*** | ***O(M*N^2)*** |

## Pruebas Realizadas

Descripción de las pruebas de tiempos de ejecución y memoria utilizada. Incluir descripción del procedimiento, las condiciones, las herramientas y recursos utilizados (librerías, computadores donde se ejecutan las pruebas, entre otros).

| | |
|---|---|
| Procesadores | Intel Core i5-1135G7 |
| Memoria RAM | 8 GB |
| Sistema Operativo | Windows 11 |

| **Entrada** | **Tiempo (s)** |
|---|---|
| small | 250.10 |
| 5 pct | 365.56 |
| 10 pct | 513.764 |
| 20 pct | 705.60 |
| 30 pct | 800.71 |
| 50 pct | 870.05 |
| 80 pct | 913.7 |
| large | 1129.2 |

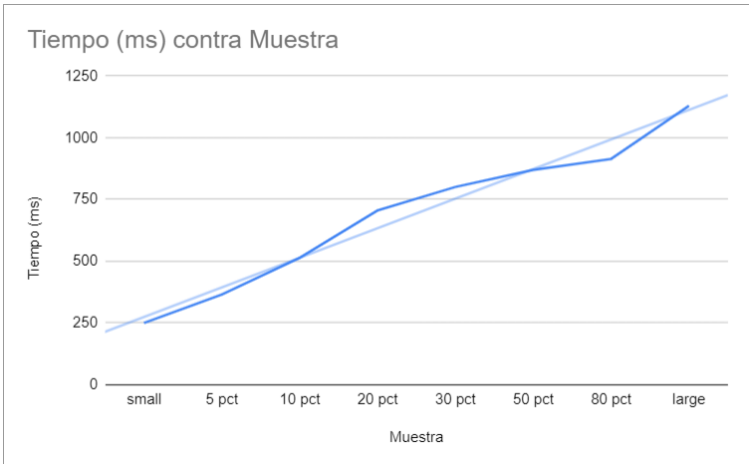
### Tablas de datos

Las tablas con la recopilación de datos de las pruebas.

| **Muestra** | **Salida** | **Tiempo (ms)** |
|---|---|---|
| small | Dato1 | 250.10 |
| 5 pct | Dato2 | 365.56 |
| 10 pct | Dato3 | 513.764 |
| 20 pct | Dato4 | 705.60 |
| 30 pct | Dato5 | 800.71 |
| 50 pct | Dato6 | 870.05 |

| | | |
|---|---|---|
| 80 pct | Dato7 | 913.7 |
| large | Dato8 | 1129.2 |

## Graficas

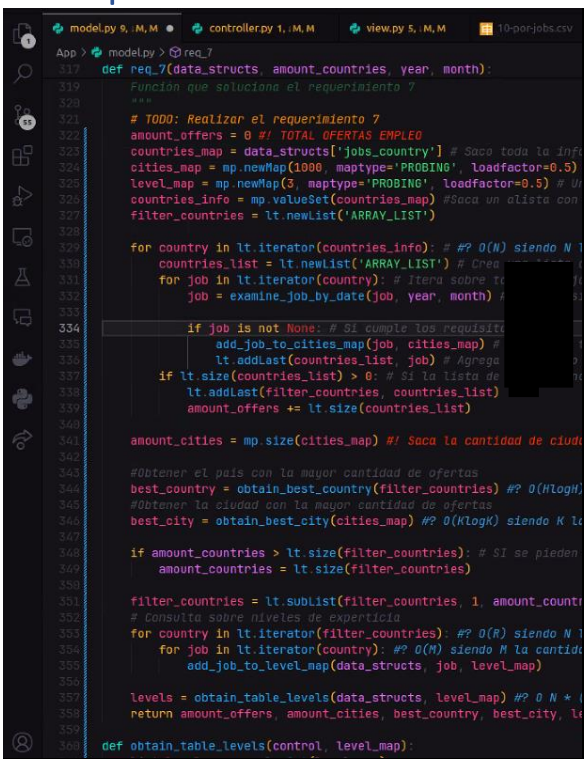



## Análisis

Se muestra en la gráfica que a medida que el tamaño de entrada aumenta, el tiempo de ejecución tiende a incrementar. Esto sugiere una relación entre el tamaño de entrada y la complejidad temporal del algoritmo.

# Requerimiento 7

## Descripción




Este código lo que hace es sacar toda la información del mapa de jobs_country luego, se crea una lista de países. Itera sobre todos los Jobs, y de acuerdo con distintas funciones implementadas examina si este cumple los requisitos de búsqueda. En caso de que cumpla los requisitos este se agrega al mapa de ciudades y se agrega el trabajo a la lista de trabajos. Posteriormente se agrega el país con mayor cantidad de ofertas y del mismo modo con la ciudad. Se genera un filtro para que si piden más elementos de los que hay solo retorne esos. Se consulta sobre niveles de experiencia y retorna la cantidad de ofertas, la cantidad de ciudades, el mejor país y ciudad y los niveles de experticia.

| | |
|---|---|
| **Entrada** | Estructuras de datos, cantidad de países, año y mes |
| **Salidas** | Número de ofertas, número de ciudades, mejor país, mejor ciudad y niveles de experticia. |
| **Implementado (Sí/No)** | Si se implementó y Karin Guerrero |

## Análisis de complejidad

Análisis de complejidad de cada uno de los pasos del algoritmo

| **Pasos** | **Complejidad** |
|---|---|
| Paso 1 | O(1) |
| Paso 2 | O(N) |
| Paso 3 | O(1) |
| Paso 4 | O(log(n) |

| | |
|---|---|
| Paso 5 | O(1) |
| Paso 6 | O(1) |
| Paso 7 | 0(NLOGN) |
| ***TOTAL*** | ***O(N)*** |

## Pruebas Realizadas

Descripción de las pruebas de tiempos de ejecución y memoria utilizada. Incluir descripción del procedimiento, las condiciones, las herramientas y recursos utilizados (librerías, computadores donde se ejecutan las pruebas, entre otros).

| | |
|---|---|
| Procesadores | Intel Core i5-1135G7 |
| Memoria RAM | 8 GB |
| Sistema Operativo | Windows 11 |

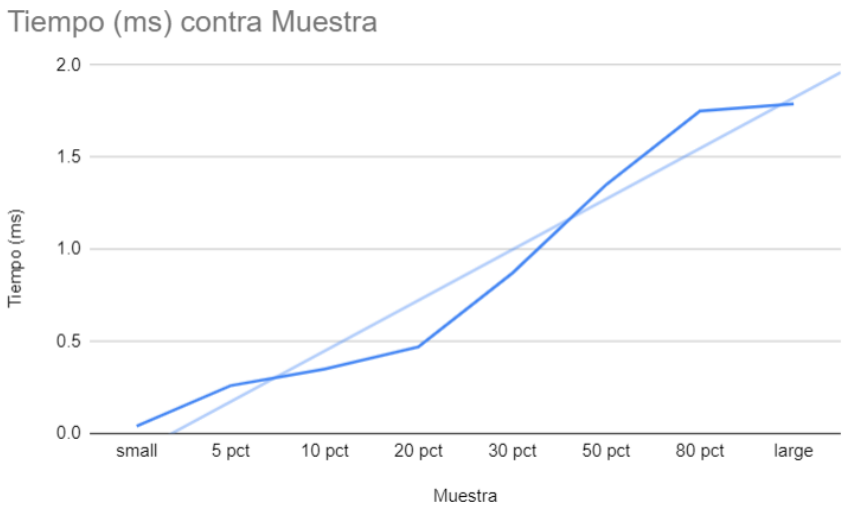
| **Entrada** | **Tiempo (s)** |
|---|---|
| small | 0.04 |
| 5 pct | 0.26 |
| 10 pct | 0.35 |
| 20 pct | 0.47 |
| 30 pct | 0.87 |
| 50 pct | 1.35 |
| 80 pct | 1.75 |
| large | 1.79 |

## Tablas de datos

Las tablas con la recopilación de datos de las pruebas.

| **Muestra** | **Salida** | **Tiempo (ms)** |
|---|---|---|
| small | Dato1 | 0.04 |
| 5 pct | Dato2 | 0.26 |
| 10 pct | Dato3 | 0.35 |
| 20 pct | Dato4 | 0.47 |
| 30 pct | Dato5 | 0.87 |
| 50 pct | Dato6 | 1.35 |
| 80 pct | Dato7 | 1.75 |
| large | Dato8 | 1.79 |

## Graficas




## Análisis

La gráfica muestra que hay una tendencia de aumento en la medida que el tamaño aumenta. Sin embargo, se observa una mayor variabilidad entre el 30 pct y el 50 pct. Además, se puede observar que el tamaño más grande es ligeramente superior al valor 80 pct que seria “large”, esta diferencia es mínima, sugiriendo una posible estabilización en la medida para tamaños muy grandes.